Captura do Histórico de Imagens

Juliana Cristina Braga, Gerald Francis Banon
INPE *INPE*
juliana@dpi.inpe.br *banon@dpi.inpe.br*

Resumo

Quando processamos uma imagem, os sistemas de processamentos não se preocupam com a captura do histórico das imagens. No entanto, a preservação do histórico pode ser muito útil. Neste trabalho, apresentaremos um sistema de processamento de imagens que captura e utiliza os históricos de imagens. Esse sistema é dividido em 4 módulos: gerador de histórico, processador de histórico, editor de histórico e tradutor de histórico. A plataforma de desenvolvimento do sistema proposto será em Python sendo o histórico escrito em metalinguagem compatível com XML.

1. Introdução

Ao processar uma imagem, os sistemas de processamento ainda não se preocupam em coletar informações sobre o histórico da imagem, ou seja, como as imagens envolvidas no processamento foram obtidas, se essas imagens são brutas ou se sofreram algum tipo de processamento, se processadas, quais as etapas de processamento foram necessárias para obtê-las. O armazenamento do histórico de uma imagem possibilitaria algumas vantagens.

a) Facilitar a execução da função de tratamento escolhido

Além da imagem em si, o armazenamento de dados descritivos sobre a própria imagem, como o seu volume (número de linhas x número de colunas), seu formato (precisão numérica para representar um valor radiométrico, 8 bits por exemplo), a maneira como ela foi obtida e a descrição do tipo de equipamento usado que produziu a imagem pode ser muito importante para um futuro processamento. Nessa descrição poderiam constar dados sobre o que representa o valor radiométrico de um ponto da imagem, a altitude do satélite no caso de um sistema de aquisição a bordo, o número de detetores usados, o tempo de aquisição entre os pontos no caso de uma observação dinâmica. O sistema de processamento de imagens deveria permitir a entrada, uma vez por todas, desses dados, e seu uso automático cada vez que fosse necessária a execução da função de processamento escolhida pelo usuário. Dessa maneira, a tarefa do usuário do sistema limitar-se-ia, à escolha da próxima função de processamento da(s) imagem(s) envolvida(s). O próprio sistema dispondo das informações úteis para o processamento evitaria incomodar o usuário com uma lista longa de perguntas.

Por exemplo, para a correção geométrica do efeito de rotação da terra de uma imagem obtida a partir de um satélite de observação da terra, o sistema de processamento deveria ser capaz de reunir no histórico todos os dados necessários para efetuar essa correção, ou seja as informações sobre a latitude do centro da imagem, a velocidade relativa e a altitude do satélite, o número de detectores e o tamanho do ponto reduzido à superfície [3].

b) Facilitar a programação de novas funções de tratamento

Numa fase de pesquisa, para se chegar a uma nova imagem que responda às expectativas do usuário, várias tentativas são geralmente necessárias. Quando o resultado final (geralmente uma imagem) é assim obtido, a seqüência dos processamentos que levaram a essa imagem constitui o algoritmo de processamento. Um sistema de processamento de imagens deveria guardar automaticamente essa seqüência. Dessa forma, quando for necessário repetir um tratamento já bem sucedido com novos dados fontes, o usuário deveria apenas trocar as referencias desses dados no algoritmo já usado e comprovado. Esse procedimento poderia também se aplicar ao caso de se querer testar a influência de um parâmetro sobre o resultado final. Neste caso o usuário deveria apenas trocar o valor do parâmetro em questão dentro do algoritmo já testado. Em outras palavras, o usuário teria a opção de programar novos tratamentos a partir de exemplos de outros tratamentos já executados.

Por exemplo, o ajuste do contraste de uma imagem, poderia ser obtido apenas trocando o vetor de parâmetros definindo a tabela de transformação que foi usada no processamento daquela imagem.

Guardando as seqüências de processamento, ao longo do tempo seria criado um grande repositório de algoritmos de processamento que poderiam ser reutilizados por outros usuários [3].

c) Auxiliar na limitação da capacidade de armazenamento

Certas imagens podem ser vistas como resultados intermediários dentro de uma cadeia de processamento; neste caso cada uma dessas imagens deveria ser automaticamente apagadas depois de terminado o último tratamento que a envolve. Se conservado apenas seu histórico, deixa-se mais espaço em memória para futuros

tratamentos. Caso necessite usar essa imagem novamente é só consultar seu histórico para saber como a mesma foi obtida e gerá-la novamente [3].

Para armazenar o histórico de uma imagem deve-se optar por uma linguagem interoperável para que qualquer sistema de tratamento consiga interpretá-la e dessa forma usufruir seus benefícios.

2. Objetivo

Tendo em vista a grande importância de armazenar o histórico do processamento de imagens, objetiva-se com esse trabalho:

- propor um modelo de um sistema de tratamento de imagens que crie e utilize o histórico de uma imagem; e
- implementar um protótipo para testar o uso do modelo proposto.

3. Metodologia

O modelo proposto deverá funcionar como um módulo adicional a ambientes de processamento (tratamento) de imagens.

3.1. Modelo

O modelo proposto possuirá 4 módulos (Figura 1).

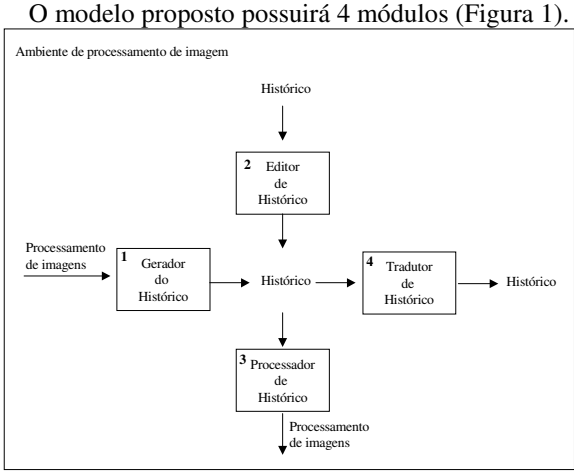

Figura 1 – Modelo proposto com seus 4 módulos.

O primeiro módulo, gerador de histórico, será responsável por capturar os passos do usuário ao processar uma imagem e registra-los no arquivo denominado histórico. O segundo módulo, editor de histórico, deverá permitir a criação de históricos ou modificações de históricos existentes. O terceiro módulo, processador de histórico, deverá ler o histórico e repetir automaticamente o processamento armazenado no mesmo. O quarto módulo, tradutor de histórico, deverá ler o histórico e a partir deste gerar algoritmos em algumas linguagens como, por exemplo, HTML, JAVA, C++, MATLAB, LATEX.

A seguir apresenta-se uma descrição dos quatro módulos.

3.1.1. Gerador de histórico. O gerador de histórico tem como funções:

1- Capturar informações sobre as imagens envolvidas em um processamento como, por exemplo, domínio, contradomínio, etc. Essas informações poderão ser extraídas automaticamente pelo gerador e outros fornecidos pelo usuário.

2- Capturar as funções de processamento que o usuário utiliza para tratar uma imagem. O conjunto dessas funções fornecerá o algoritmo do processamento. As funções deverão ser capturadas automaticamente e em paralelo com as ações do usuário.

3- Escrever as informações sobre as imagens e as funções de processamento em um arquivo denominado histórico.

Para a geração do histórico será desenvolvida uma metalinguagem. Essa metalinguagem será baseada em XML e denominada Metalinguagem para Histórico (**HML**).

O modelo será desenvolvido em Python. Sendo assim este módulo irá capturar os comandos de processamentos através do Python e transformá-los em **HML**. Essa captura de comandos será inspirada por "A service for APL2 Global Variable Manipulation" [4].

Para ilustrar o histórico de uma imagem considere o processamento mostrado na Figura 2.

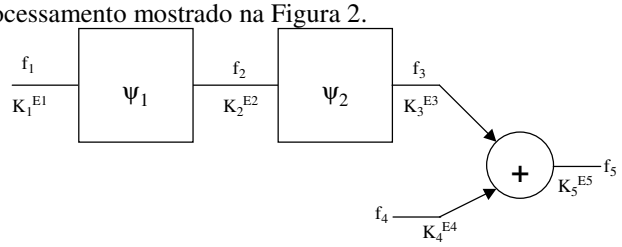

Figura 2 – Exemplo de processamento de imagem

Os históricos das imagens do processamento indicado na Figura 2 são dados a seguir:

Histórico de $f_1 = K_1 \rightarrow E_1$;
Histórico de f_2 = (histórico de f_1), ψ_2, $E_2 \rightarrow K_2$;
Histórico de f_3 = (histórico de f_2), ψ_3, $E_3 \rightarrow K_3$.
Histórico de $f_4 = E_4 \rightarrow K_4$.
Histórico de f_5 = (histórico de f_3), (histórico de f_4),+,
Onde:
ψ_i , + = operações de processamento;
f_i = imagem;
K_i = contra-domínio da imagem f_i;
E_i = domínio da imagem f_i;

A XML foi escolhida para o desenvolvimento da HML pelos seguintes motivos:

- Interoperabilidade. Tendo uma sintaxe comum, diferentes aplicações podem compartilhar os

metadados (históricos) e tratá-los uniformemente por meio de diferentes plataformas/sistemas;

- Bom suporte. Existem um número de ferramentas e pacotes de software freeware disponíveis para manipular dados em XML (parser, editores, etc).
- Flexibilidade. Permite que outras tags sejam facilmente criadas para complementar a linguagem possibilitando que esta fique cada vez mais completa.
- Garantia de qualidade. Os dados em XML podem ser checados para consistência e qualidade em obediência a certas DTDs ou XML "Schemas".

Usuários poderão editar históricos em HML usando o nódulo editor de histórico. Dessa forma, o histórico também poderá ser obtido independente do gerador de histórico.

A Figura 3 mostra como deverá ser o histórico da imagem f_5 do processamento indicado na Figura 2 escrito em HML.

```
<?xml version="1.0" encoding="UTF-8"?>
<HML>
  <HISTORICO IMAGEM="f5">
    <CABECALHO>
      <DOMINIO>K5</DOMINIO>
      <CONTRADOMINIO>E5</CONTRADOMINIO>
    </CABECALHO>
    <HISTORICO IMAGEM="f4">
      <CABECALHO>
        <DOMINIO>K4</DOMINIO>
        <CONTRADOMINIO>E4</CONTRADOMINIO>
      </CABECALHO>
    </HISTORICO>
    <HISTORICO IMAGEM="f3">
      <CABECALHO>
        <DOMINIO>K3</DOMINIO>
        <CONTRADOMINIO>E3</CONTRADOMINIO>
      </CABECALHO>
      <HISTORICO IMAGEM="f2">
        <CABECALHO>
          <DOMINIO>K2</DOMINIO>
          <CONTRADOMINIO>E2</CONTRADOMINIO>
        </CABECALHO>
        <HISTORICO IMAGEM="f1">
          <CABECALHO>
            <DOMINIO>K1</DOMINIO>
            <CONTRADOMINIO>E1</CONTRADOMINIO>
          </CABECALHO>
        </HISTORICO>
        <PROCESSAMENTO>Y1</PROCESSAMENTO>
      </HISTORICO>
      <PROCESSAMENTO>Y2</PROCESSAMENTO>
    </HISTORICO>
    <PROCESSAMENTO>+</PROCESSAMENTO
  </HISTORICO>
</HML>
```

Figura 3 – Exemplo de histórico

Observa-se na Figura 3, que a "tag" raiz é <HML> e que inserida nas "tags" <HISTORICO> devem conter as "tags" <CABECALHO>, <HISTORICO> e <PROCESSAMENTO>, sendo a primeira obrigatória e as duas últimas não obrigatórias. As "tags" inseridas em <CABECALHO> indicam contradomínio e domínio da imagem.

O código HML deverá seguir normas definidas em um "Schema" METAPI o que torna necessário o desenvolvimento do mesmo.

Pretende-se criar uma documentação de referência sobre as "tags" da HML onde serão informados quais as "tags" existentes e como utilizá-las.

3.1.2. Editor de histórico. Este módulo permitirá a edição de um novo histórico e/ou modificação de um histórico existente. O editor de histórico deverá restringir as modificações no conteúdo das tags <HISTORICO> já que o conteúdo das tags <CABECALHO> serão recalculadas automaticamente a partir das especificações dos programas nas tags <PROCESSSAMENTO>. Uma vez modificado, o histórico poderá ser processado pelo processador de histórico.

Para ilustrar, segue um exemplo da utilização do editor na modificação do histórico da imagem f_5. Suponhamos que quiséssemos gerar uma imagem f_6 a qual seria a subtração da imagem f_3 com f_4 (Figura 3). Neste momento o editor deverá verificar se essas duas imagens poderão ser subtraídas (as duas imagens deverão ter o mesmo domínio), e em caso afirmativo a modificação será feita.

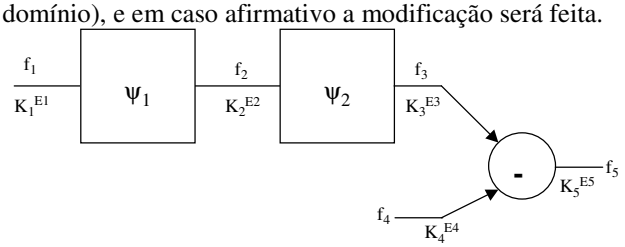

Figura 4 – Exemplo de processamento de imagem

3.1.3. Processador de histórico. A finalidade do processador de histórico é ler um histórico e executar automaticamente o processamento de imagem descrito no mesmo. O processador deverá verificar a disponibilidade das imagens envolvidas e executará apenas os processamentos necessários para regenerar as imagens faltantes.

3.1.4. Tradutor de histórico. Este módulo deverá gerar, a partir do histórico em XML, históricos em linguagens de programação como, por exemplo, HTML, C++, Java, Object Pascal, Latex, Python [8].

Escolheu-se transformar o histórico em HTML pois a leitura de um histórico no formato HML (Figura 2) pode ser confusa para usuários leigos em programação. O arquivo a ser gerado deverá ser formatado e estruturado para que o documento HTML possa ser facilmente lido. Além disso, o formato HTML, permite que o histórico seja facilmente publicado na Internet. Pretende-se criar um ambiente de publicação em que as pessoas autorizadas poderão não só consultar, mas também acrescentar históricos. Com o passar do tempo teremos uma biblioteca digital de históricos em processamento de imagens. Uma ferramenta de busca deverá ser desenvolvida para auxiliar na consulta dos históricos.

Escolheu-se transformar o histórico em outras linguagens (C++, Java, etc), para que o código contido no histórico transformado possa ser executado em outros ambientes de processamento de imagens.

3.2. Protótipo

Será utilizada para desenvolvimento do protótipo a toolbox de processamento de imagem desenvolvida na UNICAMP[9]. Essa toolbox foi implementada na linguagem Python [8]. Partindo deste fato, definiu-se Python como linguagem de programação a ser utilizada no desenvolvimento da tese.

Python é uma linguagem de programação de scripting orientada a objetos, multi-plataforma, com vasto número de bibliotecas, e que permite o rápido desenvolvimento de aplicações.

Haverá necessidade de um módulo de codificação dos históricos em Python para XML. Este módulo terá um bom suporte em Python uma vez que existe biblioteca em Python específica para aplicações em XML.

4. Resultados Esperados

Espera-se implementar parte do modelo proposto. Este deverá ser descrito detalhadamente para que futuramente (após o término do doutorado) seja todo implementado sem maiores dificuldades.

A seguir os resultados esperados em cada módulo são apresentados.

Gerador de histórico. Deverá ser implementado a HML. Associado a HML será desenvolvido o 'Shema" XML e a referencia completa da metalinguagem. O Gerador de histórico deverá ser todo implementado no protótipo.

Editor de histórico. Pretende-se implementar o editor de histórico completo.

Processador de histórico. Pretende-se implementar todo o módulo processador de histórico no protótipo.

Tradutor de histórico. Esse módulo não será todo implementado. Pretende-se implementar o transformador de histórico HTML e para isso será desenvolvido a folha de estilo XSL. A estrutura de banco de dados, ferramenta de busca e gerador de códigos compiláveis deverão ser implementadas futuramente. Apenas alguns exemplos do gerador de códigos deverão ser implementados no protótipo.

5. Referências

[1] McGrath S. XML Aplicações práticas. Rio de Janeiro: Campus, 1999.

[2] Holzer S. Desvendando XML. Rio de Janeiro: Campus, 2001.

[3] Banon, G, J, F. Implementação de um sistema de tratamento de imagens usando uma definição ampla de imagens; III Simpósio de Sensoriamento Remoto, Rio de Janeiro, RJ, 28 a 30 de novembro de 1991.

[4] Banon, G, J, F. A service for APL2 Global Variable Manipulation. Centro científico Brasília-Brasil. Relatório técnico. Maio de 1986.

[5] Especificação XML: http://www.w3.org/TR/XML/

[6] Especificação XSL: http://www.w3.org/TR/XSL/

[7] Especificação de Schemas: http://www.w3.org/Tr/xmlschema-0/

[8] Especificação do Python: http://www.python.org.

[9] Toolbox para processamento de imagens usando Python: http://www.dca.fee.unicamp.br/~alexgs/ia636